Privacy-Preserving Collaborative Sequential Pattern Mining

Justin Z. Zhan[*], *LiWu Chang*[†], *and Stan Matwin*[‡]

1 Introduction

In the modern business world, collaborative data mining becomes especially important because of the mutual benefit it brings to the collaborators. During the collaboration, each party of the collaboration needs to share its data with other parties. If the parties don't care about their data privacy, the collaboration can be easily achieved. However, if the parties don't want to disclose their private data to each other, can they still achieve the collaboration?

To use the existing data mining algorithms, all parties need to send their data to a trusted central place to conduct the mining. However in situations with privacy concerns, parties may not trust anyone, including a third party. Generic solutions for any kind of secure collaborative computing exist in the literature (e.g., [7] and [3]). However, none of the proposed generic solutions is practical in handling large-scale data sets because of the prohibitive extra cost in protecting data privacy. Therefore, practical solutions need to be developed. This need underlies the rationale for our research.

Data mining includes a number of different tasks. This paper focuses on sequential pattern mining. Specially, we study the problem of how to jointly mining sequential patterns among multiple parties while preserving data privacy of each party. To the best of our knowledge, the problem has not been investigated and is a challenge to the information security and privacy community. Our contributions are (1) to propose a new representation scheme for sequential data in order to facilitate

[*]School of Information Technology and Engineering, University of Ottawa, Canada, zhizhan@site.uottawa.ca

[†]Center for High Assurance Computer Systems, Naval Research Laboratory, USA, lchang@itd.nrl.navy.mil.

[‡]School of Information Technology and Engineering, University of Ottawa, Canada, stan@site.uottawa.ca

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Privacy-Preserving Collaborative Sequential Pattern Mining	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**12**	

mining sequential patterns among private data sets, and (2) to develop a new secure protocol for multiple parties to jointly compute the support measure of sequential patterns.

The paper is organized as follows: Section 2 discusses the related work. We then formally defines the mining sequential patterns on private data problem in Section 3. In Section 4, we describe our secure protocols. We give our conclusion in Section 5.

2 Related Work

Privacy-Preservation Multi-Party Data Mining In the early work on privacy-preserving data mining, Lindell and Pinkas [4] proposed a solution to the privacy-preserving classification problem using the oblivious transfer protocol, a powerful tool developed by the secure multi-party computation research. Vaidya and Clifton [5] proposed to use the scalar product as the basic component to tackle the problem of association rule mining in vertically partitioned data. Later, they proposed a permutation scheme to solve the K-means clustering [6] over vertically partitioned data. In [8], a secure procedure is provided to solve privacy-preserving collaborative data mining.

Sequential Pattern Mining Sequential pattern mining, introduced in [1], is concerned of inducing rules from a set of sequences of ordered items. Their method calculates the support measures of sequences by iteratively joining those sub-sequences whose supports exceed a given threshold. However, to the best of our knowledge, the issue of secure sequential pattern mining has not been studied. In this paper, we will propose a scheme to tackle the problem of privacy-preserving sequential pattern mining over the vertically partitioned data.

3 Privacy-Preserving Collaborative Sequential Pattern Mining

3.1 Background

Since its introduction in 1995 [1], the sequential pattern mining has received a great deal of attention. It is still one of the most popular pattern-discovery methods in the field of Knowledge Discovery. In the sequential pattern mining, we are given a data set D of customer transactions. Each transaction consists of the following fields: customer-id, transaction-time, and the items purchased in the transaction. No customer has more than one transaction with the same transaction-time. We do not consider quantities of items bought in a transaction: each item is a binary variable representing whether an item was bought or not. An itemset is a non-empty set of items. A sequence is an ordered list of itemsets. A customer support is a sequence s if s is contained in the customer-sequence for this customer. The support for a sequence is defined as the fraction of total customers who support this sequence. Given a data set D of customer transactions, the problem of mining

sequential patterns is to find the maximal sequences among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a sequential pattern.

3.2 Problem Definition

We consider the scenario where multiple parties, each having a private data set (denoted by D_1, D_2, $\cdots$, and D_n respectively), want to collaboratively conduct sequential pattern mining on the union of their data sets. Because they are concerned about the data privacy, neither party is willing to disclose its raw data set to others. Without loss of generality, we make the following assumptions on the data sets (the assumptions can be achieved by pre-processing the data sets D_1, D_2, $\cdots$, and D_n, and such pre-processing does not require one party to send its data set to other parties):

1. D_1, D_2, $\cdots$, and D_n are data sets, where each data set consists of the customer-id, transaction-time, and the items purchased in each transaction.

2. D_1, D_2, $\cdots$, and D_n contain the different types of items (e.g., they come from different types of markets).

3. The identity of the transactions in D_1, D_2, $\cdots$, and D_n are the same.

4. The customer-ID and customer's transaction time can be shared among the parities, but the items that a customer actually bought are confidential.

Privacy-Preserving Collaborative Sequential Pattern Mining problem: Party 1 has a private data set D_1, party 2 has a private data set D_2, $\cdots$, and party n has a private data set D_n. Data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ is the union of D_1, D_2, $\cdots$, and D_n (by vertically putting D_1, D_2, $\cdots$, and D_n together.) Let N be a set of transactions with N_k representing the kth transaction. These n parties want to conduct the sequential pattern mining on $[D_1 \cup D_2 \cup D_3 \cdots \cup D_n]$ and to find the sequential patterns with support greater than the given threshold, but they do not want to share their private data sets with each other. We say that a sequential pattern of $x_i \leq y_j$, where x_i occurs before or at the same time as y_j, has support s in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if $s\%$ of the transactions in $[D_1 \cup D_2 \cdots \cup D_n]$ contain both x_i and y_j with x_i happening before or at the same time as y_j (namely, $s\% = P(x_i \cap y_j | x_i \leq y_j)$).

3.3 Sequential Pattern Mining Procedure

The procedure of mining sequential patterns contains the following steps:
Step I: Sorting

The data set $[D_1 \cup D_2 \cdots \cup D_n]$ is sorted, with customer-id as the major key and transaction-time as the minor key. This step implicitly converts the original transaction data set into a data set of customer sequences. Since the customer-id

and the transaction time are not private among the parties, this step can be executed without using a secure protocol. As a result, transactions of a customer may appear in more than one rows, where a row contains information of a customer ID, a particular transaction time and items bought at this transaction time. For example, suppose that data sets after being sorted by their customer-id numbers are shown in Fig. 1. Then after being sorted by the transaction time, data tables of Fig. 1 will become those shown in Fig. 2.

Alice

C-ID	T-time	Items Bought
1	06/25/03	30
2	06/10/03	10, 20
2	06/20/03	9, 15
3	06/25/03	30
3	06/30/03	5, 10

Bob

C-ID	T-time	Items Bought
1	06/30/03	90
2	06/15/03	40, 60
3	06/18/03	35, 50
3	06/10/03	45, 70

Carol

C-ID	T-time	Items Bought
1	06/28/03	110
2	06/13/03	107
3	06/19/03	103
3	06/26/03	105, 106
3	06/21/03	101, 102

Figure 1. *Raw Data Sorted By Customer ID*

Alice

C-ID	T-tme	Items Bought
1	06/25/03	30
N/A		
N/A		
2	06/10/03	10, 20
N/A		
N/A		
2	06/20/03	9, 15
N/A		
N/A		
N/A		
N/A		
3	06/25/03	30
N/A		
3	06/30/03	5, 10

Bob

C-ID	T-time	Item Bought
N/A		
N/A		
1	06/30/03	90
N/A		
N/A		
2	06/15/03	40, 60
N/A		
3	06/10/03	45, 70
3	06/18/03	35, 50
N/A		
N/A		
N/A		
N/A		
N/A		

Carol

C-ID	T-time	Item Bought
N/A		
1	06/28/03	110
N/A		
N/A		
2	06/13/03	107
N/A		
N/A		
N/A		
N/A		
3	06/19/03	103
3	06/21/03	101, 102
N/A		
3	06/26/03	105, 106
N/A		

N/A: The information is not available.

Figure 2. *Raw Data Sorted By Customer ID and Transaction Time*

Step II: Mapping

Each item of a row is considered as an attribute. We map each item of a

row (i.e., an attribute) to an integer in an increasing order and repeat for all rows. Re-occurrence of an item will be mapped to the same integer. As a result, each item becomes an attribute and all attributes are binary-valued. For instance, the sequence $< B, (A, C) >$, indicating that the transaction B occurs prior to the transaction (A,C) with A and C being simultaneous events, will be mapped to integers in the order $B \rightarrow 1$, $A \rightarrow 2$, $C \rightarrow 3$, $(A, C) \rightarrow 4$. During the mapping, the corresponding transaction time will be kept. For instance, based on the sorted data set of Fig. 2, we may construct the mapping table as shown in Fig. 3. We use $'-'$ to denote the mapping in Fig. 3. For example, $'30 - 1A'$ means that we map 30 to 1A. After the mapping, the mapped data sets are shown in Fig. 4.

Alice	30 - 1A	10 - 2A	20 - 3A	(10, 20) - 4A	9 - 5A	15 - 6A	(9, 15) - 7A	5 - 8A	(5, 10) - 9A	
Bob	90 - 1B	40 - 2B	60 - 3B	(40, 60) - 4B	35 - 5B	50 - 6B	(35, 50) - 7B	45 - 8B	70 - 9B	(45, 70)- 10B
Carol	110 - 1C	107 - 2C	103 - 3C	101 - 4C	102 - 5C	(101,102)- 6C	105 - 7C	106 - 8C	(105,106) - 9C	

Note that, in Alice's dataset, item 30 and 10 are reoccurred, so we map them to the same mapped-ID.

Figure 3. *Mapping Table*

Alice

C-ID \ Mapped ID	1A		2A		3A		4A		5A		6A		7A		8A		9A	
1	1	06/25/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/10/03	1	06/10/03	1	06/10/03	1	06/20/03	1	06/20/03	1	06/20/03	0	N/A	0	N/A
3	1	06/25/03	1	06/30/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	1	06/30/03	1	06/30/03

Bob

C-ID \ Mapped ID	1B		2B		3B		4B		5B		6B		7B		8B		9B	
1	1	06/30/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/15/03	1	06/15/03	1	06/15/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
3	0	N/A	0	N/A	0	N/A	0	N/A	1	06/18/03	1	06/18/03	1	06/10/03	1	06/10/03	1	06/10/03

Carol

C-ID \ Mapped ID	1C		2C		3C		4C		5C		6C		7C		8C		9C	
1	1	06/28/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
2	0	N/A	1	06/13/03	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
3	0	N/A	0	N/A	1	06/19/03	1	06/21/03	1	06/21/03	1	06/21/03	1	06/26/03	1	06/26/03	1	06/26/03

N/A : The information is not available.

Figure 4. *Data After Being Mapped*

Step III: Mining

Our mining procedure will be based on mapped data set after the mapping

step. The general sequential pattern mining procedure contains multiple passes over the data. In each pass, we start with a seed set of large sequences, where a large sequence refers to a sequence whose itemsets all satisfy the minimum support. We utilize the seed set for generating new potentially large sequences, called candidate sequences. We find the support for these candidate sequences during the pass over the data. At the end of each pass, we determine which of the candidate sequences are actually large. These large candidates become the seed for the next pass. The following is the procedure for mining sequential patterns on $[D_1 \cup D_2 \cdots \cup D_n]$.

1. L_1 = large 1-sequence
2. for (k = 2; $L_{k-1} \neq \phi$; k++) **do**{
3. C_k = **apriori-generate**(L_{k-1})
4. for all candidates $c \in C_k$ **do** {
5. **Compute** *c.count*

(Section 3.4 will show how to compute this count on private data)

6. $L_k = L_k \cup$ c | *c.count* $\geq$ *minsup*
7. end
8. end
9. Return UL_k

where L_k stands for a sequence with k itemsets and C_k stands for the collection of candidate k-sequences. The procedure **apriori-generate** is described as follows:

Step 1: join L_{k-1} with L_{k-1}:

1. insert into C_k
2. select $p.litemset_1$, $\cdots$, $p.litemset_{k-1}$, $q.litemset_{k-1}$, where $p.litemset_1 = q.litemset_1$, $\cdots$, $p.litemset_{k-2} = q.litemset_{k-2}$
3. from L_{k-1} p, L_{k-1} q.

Step 2: delete all sequences $c \in C_k$ such that some (k-1)-subsequence of c is not in L_{k-1}.

Step IV: Maximization

Having found the set of all large sequences S in the sequence phase, we provide the following procedure to find the maximal sequences.

1. for (k = m; $k \leq 1$; k- -) **do**
2. for each k-sequence s_k **do**
3. **Delete** all subsequences of s_k from S

Step V: Converting

The items in the final large sequences are converted back to the original item representation before the mapping step. For example, if 1A belongs to some large sequential pattern, then 1A will be converted to item 30, according to the mapping table, in the final large sequential patterns.

3.4 How to compute *c.count*

To compute *c.count*, in other words, to compute the support for some candidate pattern (e.g., $P(x_i \cap y_i \cap z_i | x_i \geq y_i \geq z_i)$), we need to consider two aspects: one is to deal with the condition part where z_i occurs before y_i and both of them occur before x_i; the other is to compute the actual counts for this sequential pattern.

If all the candidates belong to just one party, then *c.count*, which refers to the frequency counts for candidates, can be computed by this party alone since this party has all the information. However, if the candidates belong to different parties, it is a non-trivial problem to conduct the joint frequency counts while protecting privacy of data. We provide the following steps to conduct this cross-parties' computation.

Step I: Vector construction
The parities construct vectors for their own attributes (mapped-id). Suppose we want to compute the *c.count* for $2A \geq 2B \geq 6C$ in Fig. 4. We construct three vectors: 2A, 2B and 6C as in Fig. 5.

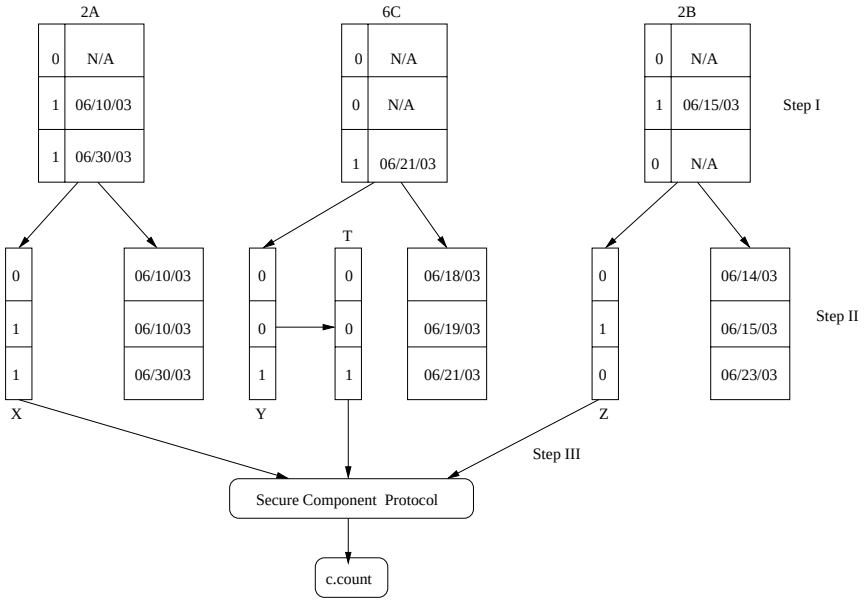

Figure 5. *An Protocol To Compute c.count*

Step II: Transaction time comparison

1. All the parties randomly generate a set of resonable[1] transaction time for

[1]Transaction time in the parties's data sets is usually distributed over some periods. By reasonable, we mean the randomly generated time should fall into these periods. If a party fails to do so, it would let other parties to easily make a correct guess of its entry values.

entries in the vector where their values are 0. The purpose of this step is to prevent one party from correctly guessing other parties' data.

2. Randomly select one party who will receive the transaction time vectors from the other two parties. For instance, in our example, Carol is selected to receive transaction time vectors for $2A$ and $2B$ from Alice and Bob.

3. Carol then compares the transaction time of each entry of $2A$, $2B$, and $6C$. She makes a temporary vector T. If the transaction time does not satisfy the requirement of $2A \geq 2B \geq 6C$, she sets the corresponding entries of T to 0's; otherwise, she copies the original values in 6C to T (Fig. 5).

In the data sets, if the item value is 0, then there is no transaction time associated with it. Therefore, if one party (e.g., Alice) sends her actual transaction time vector to other parties, then other parties can immediately know the values of those entries whose transaction time data are not present. Thus, it leads to an information leak. To enhance the data privacy, instead of sending the actual transaction time vector to other parties, Alice randomly generates a set of random transaction time for those entries that have values 0's. In other words, Alice adds some random noise to the transaction time vector. By doing so, other parties (e.g., Bob) cannot directly know which transaction does not occur. If Bob takes a random guess for each entry value based on received transaction time vector, he has the probability of 0.5 to make a correct guess provided that he has no other additional known information. With this random transaction time generation method, we can enhance the data privacy.

Step III: Compute *c.count*

After they compare their transaction time for the candidate vectors (e.g., 2A, 2B and T), they apply the secure protocols, which will be discussed in Section 4, on the value vectors to obtain the *c.count*. For example, to obtain *c.count* for $2A'$, $2B'$, and $6C'$ in Fig. 5, they need to compute $\sum_{i=1}^{N} X[i] \cdot Z[i] \cdot T[i] = \sum_{i=1}^{3} X[i] \cdot Z[i] \cdot T[i] = 0$, where N is the total number of values in each vector.

4 Secure Protocols

How two or multiple parties jointly compute *c.count* without revealing their raw data to each other is the challenge that we want to address. We propose two protocols to tackle this challenge: One is for two parties to conduct the multiplication operation; the other, with the first protocol as the basis, is designed for the secure multi-party product operation.

4.1 Introducing The Commodity Server

For performance reasons, we use an extra server, the commodity server [2] in our protocol. The parties could send requests to the commodity server and receive data (called *commodities*) from the server, but the commodities must be independent of

the parties' private data. The purpose of the commodities is to help the parties conduct the desired computations.

The commodity server is semi-trusted in the following senses: (1) It is not trusted, cannot derive the private information of the data from the parties, and cannot learn the computation result. (2) It will not collude with all the parties. (3) It follows the protocol correctly. Because of these characteristics, we say that it is a semi-trusted party. In the real world, finding such a semi-trusted party is much easier than finding a trusted party.

4.2 Component Protocols

Let's first consider the case of two parties where $n = 2$ (more general cases where $n \geq 3$ will be discussed later). Alice has a vector X and Bob has a vector Y. Both vectors have N elements. Alice and Bob want to compute the product between X and Y such that Alice gets $\sum_{i=1}^{N} U_x[i]$ and Bob gets $\sum_{i=1}^{N} U_y[i]$, where $\sum_{i=1}^{N} U_x[i] + \sum_{i=1}^{N} U_y[i] = \sum_{i=1}^{N} X[i] \cdot Y[i] = X \cdot Y$. $U_y[i]$ and $U_x[i]$ are random numbers. Namely, the scalar product of X and Y is divided into two secret pieces, with one piece going to Alice and the other going to Bob. We assume that random numbers are generated from the integer domain.

Protocol 1. *(Secure Two-party Protocol)*

1. The Commodity Server generates two random numbers $R_x[1]$ and $R_y[1]$, and lets $r_x[1] + r_y[1] = R_x[1] \cdot R_y[1]$, where $r_x[1]$ (or $r_y[1]$) is a randomly generated number. Then the server sends $(R_x[1], r_x[1])$ to Alice, and $(R_y[1], r_y[1])$ to Bob.

2. Alice sends $\hat{X}[1] = X[1] + R_x[1]$ to Bob.

3. Bob sends $\hat{Y}[1] = Y[1] + R_y[1]$ to Alice.

4. Bob generates a random number $U_y[1]$, and computes $\hat{X}[1] \cdot Y[1] + (r_y[1] - U_y[1])$, then sends the result to Alice.

5. Alice computes $(\hat{X}[1] \cdot Y[1] + (r_y[1] - U_y[1])) - (R_x[1] \cdot \hat{Y}[1]) + r_x[1] = X[1] \cdot Y[1] - U_y[1] + (r_y[1] - R_x[1] \cdot R_y[1] + r_x[1]) = X[1] \cdot Y[1] - U_y[1] = U_x[1]$.

6. Repeat step 1-5 to compute $X[i] \cdot Y[i]$ for $i \in [2, N]$. Alice then gets $\sum_{i=1}^{N} U_x[i]$ and Bob gets $\sum_{i=1}^{N} U_y[i]$.

Theorem 1. *Protocol 1 is secure such that Alice cannot learn Y and Bob cannot learn X either.*

Proof. The number $\hat{X}[i] = X[i] + R_x[i]$ is all what Bob gets. Because of the randomness and the secrecy of $R_x[i]$, Bob cannot find out $X[i]$. According to the protocol, Alice gets (1) $\hat{Y}[i] = Y[i] + R_y[i]$, (2) $Z[i] = \hat{X}[i] \cdot Y[i] + (r_y[i] - U_y[i])$, and (3) $r_x[i]$, $R_x[i]$, where $r_x[i] + r_y[i] = R_x[i] \cdot R_y[i]$. We will show that for any arbitrary

$Y'[i]$, there exists $r'_y[i]$, $R'_y[i]$ and $U'_y[i]$ that satisfies the above equations. Assume $Y'[i]$ is an arbitrary number. Let $R'_y[i] = \hat{Y}[i] - Y'[i]$, $r'_y[i] = R_x[i] \cdot R_y[i] - r_x[i]$, and $U'_y[i] = \hat{X}[i] \cdot Y'[i] + r'_y[i]$. Therefore, Alice has (1) $\hat{Y}[i] = Y'[i] + R'_y[i]$, (2) $Z[i] = \hat{X}[i] \cdot Y'[i] + (r'_y[i] - U'_y[i])$ and (3) $r_x[i]$, $R_x[i]$, where $r_x[i] + r'_y[i] = R_x[i] \cdot R'_y[i]$. Thus, from what Alice learns, there exists infinite possible values for $Y[i]$. Therefore, Alice cannot know Y and neither can Bob know X. □

We have discussed our protocol of secure number product for two parties. Next, we will consider the protocol for securely computing the number product for multiple parties. For simplicity, we only describe the protocol when $n = 3$. The protocols for the cases when $n > 3$ can be similarly derived.

Protocol 2. *(Secure Multi-party Protocol)*

Step I

1. Alice generates a random number $R_x[1]$.
2. Bob generates two random numbers $R'_y[1]$ and $R''_y[1]$.
3. Carol generates a random number $R_z[1]$.

Step II

1. Carol computes $Z[1] + R_z[1]$ and sends it to Bob.
2. Bob computes $Y[1] + R_y[1]$ and sends it to Alice.
3. Alice computes $T[1] = (X[1] + R_x[1]) * (Y[1] + R_y[1]) * (Z[1] + R_z[1])$.
4. Alice and Bob use Protocol 1 to compute $X[1] \cdot R_y[1]$, $R_x[1] \cdot Y[1]$, $R_x[1] \cdot R_y[1]$, $X[1] \cdot Y[1]$ and $R_x[1] \cdot R'_y[1]$. Then Alice obtains $U_x[1]$, $U_x[2]$, $U_x[3]$, $U_x[4]$ and $U_x[5]$ and Bob obtains $U_y[1]$, $U_y[2]$, $U_y[3]$, $U_y[4]$ and $U_y[5]$.
5. Alice sends $(U_x[1] + U_x[2] + U_x[3])$ and $(U_x[4] + U_x[1] + U_x[2] + U_x[5])$ to Carol.
6. Bob sends $(U_y[1] + U_y[2] + U_y[3])$ and $(U_y[4] + U_y[1] + U_y[2] + U_y[5])$ to Carol.
7. Carol computes
   $T_1[1] = (X[1] \cdot R_y[1] + R_x[1] \cdot Y[1] + R_x[1] \cdot R_y[1]) \cdot Z[1]$, and
   $T_2[1] = (X[1] \cdot Y[1] + X[1] \cdot R_y[1] + R_x[1] \cdot Y[1] + R_x[1] \cdot R'_y[1]) \cdot R_z[1]$.
8. Alice and Carol use Protocol 1 to compute $R_x[1] \cdot R_z[1]$ and send the values they obtained from the protocol to Bob.
9. Bob computes $T_3[1] = R''_y[1] \cdot R_x[1] \cdot R_z[1]$.

Step III

1. Repeat the Step I and Step II to compute $T[i]$, $T_1[i]$, $T_2[i]$ and $T_3[i]$ for $i \in [2, N]$.

2. Alice then gets
   $[a] = \sum_{i=1}^{N} T[i] = \sum_{i=1}^{N} (X[i] + R_x[i]) * (Y[i] + R_y[i]) * (Z[i] + R_z[i])$.

3. Bob gets
   $[b] = \sum_{i=1}^{N} T_3[i] = \sum_{i=1}^{N} (Y[i] + R_y[i]) * (Z[i] + R_z[i])$.

4. Carol gets
   $[c] = \sum_{i=1}^{N} T_1[i] = \sum_{i=1}^{N} (X[i] \cdot R_y[i] + R_x[i] \cdot Y[i] + R_x[i] \cdot R_y[i]) \cdot Z[i]$, and
   $[d] = \sum_{i=1}^{N} T_2[i] = \sum_{i=1}^{N} (X[i] \cdot Y[i] + X[i] \cdot R_y[i] + R_x[i] \cdot Y[i] + R_x[i] \cdot R'_y[i]) \cdot R_z[i]$.
   Note that $\sum_{i=1}^{N} X[i] \cdot Y[i] \cdot Z[i] = \sum_{i=1}^{N} T_0[i] = [a] - [b] - [c] - [d]$.

Theorem 2. *Protocol 2 is secure such that Alice cannot learn Y and Z, Bob cannot learn X and Z, and Carol cannot learn X and Y.*

Proof. According to the protocol, Alice obtains (1) $(Y[i] + R_y[i])$, and (2)$(Z[i] + R_z[i])$. Bob gets (1)$R_x[i] \cdot R_z[i]$. Carol gets (1)$(X[i] \cdot R_y[i] + R_x[i] \cdot Y[i] + R_x[i] \cdot R_y[i])$ and (2)$(X[i] \cdot Y[i] + X[i] \cdot R_y[i] + R_x[i] \cdot Y[i] + R_x[i] \cdot R'_y[i])$.

Since $R_x[i]$, $R_y[i](= (R'_y[i] + R''_y[i]))$ and $R_z[i]$ are arbitrary random numbers. From what Alice learns, there exists infinite possible values for $Y[i]$ and $Z[i]$. From what Bob learns, there also exists infinite possible values for $Z[i]$. From what Carol learns, there still exists infinite possible values for $X[i]$ and $Y[i]$.

Therefore, Alice cannot learn Y and Z, Bob cannot learn X and Z, and Carol cannot learn X and Y either.

□

5 Conclusion

In this paper, we considered the problem of mining sequential patterns with multiple data sets. In order to effectively compute the support measure of a sequence, we proposed a mapping scheme which converts data sequences into a binary matrix representation for frequency count of patterns. We presented and analyzed our proposed secure protocol designed for multiple parties to jointly conduct sequential pattern mining. Within this secure protocol, we introduced the multiple-party number product computation as the basic building block. In our future work, we will extend our method to deal with other data mining algorithms.

Bibliography

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[2] D. Beaver. Commodity-based cryptography (extended abstract). In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, El Paso, TX USA, May 4-6 1997.

[3] O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il/home/oded/ public_html/foc.html, 1998.

[4] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Advances in Cryptology - CRYPTO '00*, 1880 of Lecture Notes in Computer Science. Spinger-Verlag:36–54, 2000.

[5] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 23-26 2002.

[6] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Ddata Mining*, 2003.

[7] A.C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[8] Z. Zhan and L. Chang. Privacy-preserving collaborative data mining. In *Workshop on Foundation and New Direction of Data Mining at The 2003 IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, Florida, USA, November 19 2003.